METHOD AND DEVICE FOR TRANSMITTING DATA ON A SINGLE LINE, IN PARTICULAR FOR TRANSMITTING DATA ON A BUS WITH MINIMIZATION OF THE BUS SWITCHING ACTIVITY, AND CORRESPONDING COMPUTER PRODUCT

FIELD OF THE INVENTION

The present invention relates to a method and a device for transmitting data on a single line, and to a corresponding computer product.

In particular, the present invention finds advantageous but non-exclusive application for transmitting data on a bus with minimization of the bus switching activity, to which the following description will refer explicitly, without this implying any loss of generality.

BACKGROUND OF THE INVENTION

As is known, the switching activity on a wide bus or a narrow bus is responsible for a non-negligible absorption of electrical energy by the bus.

A bus can in fact be considered as a set of transmission lines, to each of which there is associated a parasitic capacitance, which, at each switching of the corresponding line, *i.e.*, at each transition $0\rightarrow1$ or $1\rightarrow0$ of the signals travelling thereon, must be charged or discharged, with consequent absorption of electrical energy.

Reduction of bus switching activity is hence an issue to which there has been dedicated, and there continues to be dedicated, an intense and extensive research activity aimed at reducing the absorption of electrical energy and, more in general, at preventing the adverse phenomena linked to the capacitive behaviour of the bus.

A technique that is widely used for reducing bus switching activity consists in encoding the data stream to be transmitted on the bus using an invertible encoding law, *i.e.*, one that can be decoded.

The encoding technique known as "bus-inverted" technique is currently the one most widely used for reducing bus switching activity, both on account of its ease of implementation, and on account of the good performance achieved, especially where the total number of lines of the bus is small. This encoding further proves to be usable also on asynchronous buses.

For a more detailed review of the subject, the following works may be usefully consulted: *Adaptive Bus Encoding Technique for Switching Activity Reduced Data Transfer over Wide System Buses*, Claudia Kretzschmar, Robert Siegmund and Dietmar Müller, International Workshop - Power and Timing Modelling, Optimization and Simulation (PATMOS2000) Goettingen (D), September 13-15, 2000; and *Architectures and Synthesis Algorithms for Power-Efficient Bus Interfaces*, I. Benini, A. Macii, E. Macii, M. Poncino, R. Scarsi, IEEE Transactions on CAD, Vol. 19, No. 9, September, 2000, pp. 969-980.

BRIEF SUMMARY OF THE INVENTION

The improved solution proposed by the present applicant for reducing bus switching activity includes the transmission of the sorting pattern P_t used by a swap operator at the transmission end so that the data transmitted may to be recovered properly at the reception end, the transmission being made using additional lines the additional switching activity of which degrades the total performance of the system.

The aim of the present invention is thus to provide an improved solution that will enable a significant reduction in the number of additional lines necessary for transmitting the sorting pattern.

The above aim is achieved by the present invention in so far as it relates to a method and a device for transmitting data on a single line, to a method and to a device for transmitting data on a bus with minimization of the bus switching activity, and to the corresponding computer products, as defined in Claims 1, 5, 9, 10, 14 and 18, respectively.

Basically, the idea underlying the present invention is to use two functionally identical finite state machines, one at the transmission end and one at the reception end, in which each finite state machine has a number of internal states equal to the number of possible sorting patterns and in which each internal state is uniquely associated to a respective sorting pattern.

Finite state machines at the transmission end and at the reception end receive the same clock signal so as to be synchronized with one another, *i.e.*, at each time their internal states coincide. For this reason, when the internal state of the finite state machine at the transmission end coincides with the sorting pattern to be transmitted, this generates a synchronization signal that is transmitted to the finite state machine at the reception end using a single transmission line. The finite state machine at the reception end thus determines the sorting pattern associated to its internal state at the moment of reception of the synchronization signal, a sorting pattern which, thanks to the fact that the internal states of the finite state machines are the same at each instant of time, is exactly identical to the one transmitted.

The present invention thus makes it possible to reduce to one the additional transmission lines necessary for transmitting the sorting pattern, with consequent drastic reduction in the additional switching activity due precisely to the additional lines.

According to a further aspect of the present invention, the additional switching activity can be further reduced by causing the coincidence between the internal state of the finite state machine at the transmission end and the sorting pattern to be transmitted to be signalled to the finite state machine at the reception end by simply complementing the synchronization signal, which involves a single switching on the bus (single switching edge), instead of by generating a synchronization pulse, which involves two switchings on the bus (double switching edge of the pulse, a leading one and a trailing one). Even though it is not optimal, the latter technique could, however, be in any case used.

BRIEF DESCRIPTION OF THE DRAWINGS

For a better understanding of the present invention there is now described a preferred embodiment, provided purely by way of non-limiting example and with reference to the attached drawings, in which:

Figure 1 illustrates, in general terms, the operating criteria of a swap operator that can be used in the framework of the invention;

Figure 2 is a block diagram that illustrates the transmission of the information on a bus;

Figures 3 and 4 are block diagrams that illustrate two implementations of the swap operator of Figure 1;

Figures 5 and 6 are block diagrams that illustrate the working principle of a first embodiment of the present invention;

Figures 7a and 7b illustrate, in a comparative manner, quantities corresponding to the first embodiment of the present invention;

Figure 8 is a block diagram that illustrates a possible variant of the first embodiment of the present invention at the transmission end; and

Figures 9 and 10 are block diagrams that illustrate the working principle of a second embodiment of the present invention.

DETAILED DESCRIPTION

One solution is based upon the use of a swap operator operating according to the criteria illustrated in Figure 1, in which $b(t)$ designates the input data stream of the swap operator, designated by 1, at the time t, $B(t)$ designates the corresponding output data stream of the swap operator 1 at the same time, and P_t designates the sorting pattern implemented by the swap operator 1. Furthermore, if the input data and the output data of the swap operator are represented by $N = 2n$ bits, the sorting pattern P_t is represented by N digits, each consisting of n bits, and will be able to assume $N!$ (N factorial) distinct values.

See for example an improved solution for reducing the bus switching activity is proposed in the European patent application No. 02425456.7, filed on July 10, 2002 by the present applicant and, broadly speaking, substantially consisting in switching the input lines of the bus using a sorting pattern that is chosen so as to minimize the switching activity between the current transmission and the previous transmission.

In the specific case represented in Figure 1, the input datum at time t is $b(t)$ = 0110, whilst the output datum at the same time is $B(t)$ = 1001.

Assuming then that to each of the bits of the input and output data there is assigned a numbering corresponding to its position, for example 0, 1, 2, 3 proceeding from the top down, in the example represented in Figure 1, the sorting pattern is therefore P_t = {2-3-0-1}, the meaning of which is as follows:

- the input bit having position 0 becomes the output bit having position 2;
- the input bit having position 1 becomes the output bit having position 3;
- the input bit having position 2 becomes the output bit having position 0; and
- the input bit having position 3 becomes the output bit having position 1.

In more general terms, it may be said that the swap operator can be mathematically modelled via a swap operator S that sets in relation the input datum, the output datum and the sorting pattern, *i.e.*, an operator for which there applies a relation of the type:

$$B(t) = S(b(t), P_t)$$

The swap operator in general admits of an inverse function S^{-1}, whereby:

$$b(t) = S^{-1}(B(t), P_t)$$

The direct and inverse swapping operations may be performed using the same function by applying two different sorting patterns connected by the bi-unique relation:

$$b(t) = S^{-1}(B(t), P_t) = S(B(t), P_t^{-1})$$

From what has just been said, it is possible to understand that, with a given input data stream *b(t)*, it is possible to make *N!* attempts to measure the switching activity between the previous output data *B(t-1)* and a given current attempt function $\tilde{B}(t)$ calculated by varying an attempt sorting pattern $\tilde{p}$.

For example, the measurement of the switching activity *SA* can be expressed by resorting to the Hamming distance applied to the function of exclusive OR between *B(t-1)* and $\tilde{B}(t)$, namely:

$$\min_{\tilde{p}} SA(\tilde{p}) = \min_{\tilde{p}} H\left[B(t-1) \oplus \tilde{B}(t)\right] \Rightarrow P_t$$

When the attempt sorting pattern $\tilde{p}$ becomes equal to the optimal one, *i.e.*, P_t, then we have B(t) = $\tilde{B}(t)$.

There are various degrees of freedom for the choice of $\tilde{B}(t)$, but this depends specifically upon the current input datum *b(t)* and upon the current sorting pattern P_t.

In what follows some examples of attempt functions are presented that are suitable for being used in a particularly advantageous way both on account of their simple form and on account of the possibility of enabling a easy decoding:

I. $\tilde{B}(t) = S(b(t), \tilde{p})$

II. $\tilde{B}(t) = S(b(t), \tilde{p}) \oplus S^{-1}(b(t\text{-}1), \tilde{p})$

III. $\tilde{B}(t) = S(b(t), \tilde{p}) \oplus S^{-1}(B(t\text{-}1), \tilde{p})$

The tests conducted by the present applicant, simulating the transmission, on a 32-bit bus, of files of a different type (Latex, Spice, GCC, JPEG, MP3 and AVI) show - with reference to a cluster depth *M* equal to four and with the use of all 24 possible sorting patterns - a reduction in the switching activity *SA* by

adopting the bus inverted technique and considering also a switching activity on the additional lines used for transmission of the sorting pattern P_t, of between 0% and 10.64%. In particular, using Function I, the reduction was between 2.74% and 14.56%; using Function II the reduction found was between 3.3% and 17.72%; and using Function III the reduction was between 15.5% and 23.16%.

It is also interesting to note how an improvement in terms of performance may depend upon the width of the bus. For example, comparing the results obtainable using the bus inverted technique with the results obtainable using Function III considered previously for a Spice file, it may be noted that for a bus having 8 lines the two solutions yield results that are practically equivalent. On buses having 32 and 40 lines the performance that can be achieved using Function III leads to a reduction in the switching activity that is practically twice what can be achieved using the bus inverted technique. In the case of a 64-line bus, Function III seen previously leads to a reduction in the switching activity that is almost three times what can be achieved using the bus inverted technique.

Returning again to the transmission of the output bits of the swap operator on the bus, the transmitter responsible for carrying out said function generates the output after making *N!* attempts, using the sorting pattern P_t that has given rise to the minimum switching activity SA.

However, if *N* is a correspondingly large number, the number of attempts required becomes rather high, and this forces the transmitter in question to operate at a speed much greater than that of the bus, *i.e.*, with a clock frequency much higher than the clock frequency at which the data are sent onto the bus.

It follows that this technique, which in itself is effective, can be used in an altogether satisfactory way only with buses having a rather slow clock.

In order to overcome this drawback it is possible to operate with a subset of the sorting patterns allowed and to use a parallel processing, which increases the area of silicon occupied by the transmitter. Furthermore, if the value

of N is high, the number of bits that represent the sorting pattern P_t increases exponentially.

If a subset of the allowed sorting patterns chosen by analysing the mean traffic and selecting the best sorting pattern is considered, there is obtained a reduction in the number of additional lines. At the same time, the fact of reducing the sorting patterns allowed with respect to an ideal solution means that there is obtained a reduction in the coverage, with a consequent degradation of the performance in overall terms.

More specifically, the results of the simulations carried out by the present applicant show that the gain in terms of reduction in the switching activity SA cannot be appreciable, should a subset of sorting patterns be chosen without following a precise criterion.

If the choice of the allowed sorting patterns is governed via test files and the recurrences of the best sorting patterns starting from the original algorithm are measured, different results are obtained.

For example, with $N = 32$ and $M = 4$, comparing the performance of sorting patterns of 4, 8, 16, and 24 bits (complete) on a Spice file, it may be noted that the best performance is obtained using 16 sorting patterns.

With reference to the results seen above, it is again possible to note that the data referred to previously with reference to Functions I, II and III can be further improved with reference to certain files using the sixteen best sorting patterns via analysis of the mean traffic. This is found to be true even though for other files the fact of resorting to this solution does not lead to an improvement, but rather to a slight deterioration in performance with respect to the performance cited previously.

In any case, the foregoing applies even though Function III continues to yield by far the best performance in terms of reduction of switching activity.

As has been said previously, when the bus is formed by a number of lines N that is relatively large, the number of attempts made for determining the

optimal sorting pattern before making the transmission becomes rather high, which forces the transmitter to operate at a speed much greater than that of the bus, *i.e.*, with a clock frequency that is much higher than the clock frequency with which the bus input data vary.

In order to overcome this drawback, it is also possible to divide the bus into a subset of narrower buses each formed by M lines, with M much smaller than N. The narrower buses are known as "clusters", and M is the cluster depth. For the various clusters the same sorting pattern is used, which is selected so as to minimize the total switching activity on the wide bus, *i.e.*, the switching activity on all the N/M clusters.

Of course, in order to be able to recover the data properly at the reception end, the sorting pattern P_t used by the swap operator at the transmission end must necessarily be transmitted to the reception end using additional lines.

Assuming that the cluster depth M is a power of two, the number of lines necessary for transmitting a sorting pattern of a cluster with depth M are $M \cdot \log_2 M$. Such a large number of lines is in actual fact redundant in so far as it is necessary to represent just $M!$ different states. During transmission, the sorting patterns can thus be compressed on the number of lines strictly necessary for representing $M!$ different states. It follows that the number of additional lines necessary for said purpose is given by the first integer greater than $\log_2 M!$.

Consequently, in transmission a compression module must be provided that carries out compression of the bits of the sorting pattern P_t and in reception a decompression module must be provided that carries out decompression of the bits transmitted for reconstructing the sorting pattern P_t.

The modules that implement the compression and decompression functions are configurable as simple combinatorial logic networks which are designed to implement a truth table and which do not include flip-flops.

Figure 2 is a schematic illustration, in the form of a functional block diagram, of the logic used for transmitting and receiving the information

corresponding to the sorting pattern transmitted on specially provided additional lines of the bus.

In the diagram of Figure 2, TX and RX designate respectively, the transmission end and the reception end with respect to the bus, which is designated by 2. The reference number 3 designates the compression module which, at the transmission end, compresses the bits of the sorting pattern on $M \cdot \log_2 M$ bits necessary for transmitting said sorting pattern on the number of lines identified by the integer greater than $\log_2 M!$; the reference number 4 designates the decompression module which, at the reception end, decompresses the $M \cdot \log_2 M$ bits transmitted for reconstructing the sorting pattern P_t; and the numbers 5 and 6 designate the compression module and, respectively, the decompression module that perform the compression and decompression functions, respectively in the transmission step and in the reception step with respect to the additional lines of the bus 2.

For transmitting the sorting pattern on the additional lines, it is possible to use advantageously the above-mentioned bus inverted technique, which, where the additional lines number just a few, enables a reduction in the switching activity to be obtained close to 60-70% as compared to the total switching activity.

The diagram of Figure 3 illustrates a simple implementation of the swap operator 1 of Figure 1. The implementation illustrated is designed to operate on just one cluster and is made up of M multiplexers (where in the example illustrated here they number four). The multiplexers receive the input data stream $b(t)$ and implement the switching driven by the sorting pattern so as to give rise to the data stream $B(t)$.

The diagram of Figure 4 illustrates, instead, how it is possible to provide a structure capable of operating on a bus with a number of lines equal to M, where it is assumed that M is quite a high number. In this case, the solution described envisages use of K swap operators of the same type as the one

described previously, where $K = (N/M) - 1$. In other words, this solution corresponds to the fact of having divided ideally a wide bus with M lines into N narrow buses, each of which is formed by M/N lines. This operating mode means that the attempts at selection of the optimal sorting pattern to be implemented on the various narrow buses are much less numerous than the ones that should be implemented on the wide bus. As has already been said, it is possible, in particular, to cause a single sorting pattern to be finally used for all the narrow buses chosen as the sorting pattern that minimizes the total bus switching activity.

Furthermore, as described previously, the direct and inverse swapping operation can be performed using the same swap operator but with different input sorting patterns. If P_t represents the direct swapping operation, there always exists a sorting pattern $P_x = P_t^{-1}$ that performs the inverse swapping function:

$$S(b(t), P_t) = S^{-1}(b(t), P_t^{-1}) = S^{-1}(b(t), P_x), \forall b(t)$$

P_x and P_t are linked by a bi-unique relation, so that it is possible to use a combinatorial network to obtain P_x starting from P_t.

For example the table presented below gives different values of P_t and P_x when $M = 4$, in order to clarify how the combinatorial network may be created.

P_t	P_x
1-2-3-0	3-0-1-2
1-0-2-3	1-0-2-3
2-1-3-0	3-1-0-2
0-1-3-2	0-1-3-2
2-1-0-3	2-1-0-3
1-3-0-2	2-0-3-1

A module of this sort, which can be defined simply as a sorting pattern converter can again be implemented, as possible alternative, as a look-up table.

Figures 5 and 6 illustrate the circuit architecture, at the transmission end and at the reception end, with which synchronization of the sorting pattern is obtained according to a first embodiment of the present invention.

In particular, Figure 5 illustrates a data source 10, consisting for example of a digital processor, which supplies the data to be transmitted on the bus 2; a transmitter 11 having the function of encoding the data supplied by the data source 10 in the way previously described so as to reduce the bus switching activity; a FIFO (First In First Out) register 12 having the purpose of accumulating the data supplied by the transmitter 11 before their introduction onto the bus 2, for the reasons that will be clarified in what follows; and a finite state machine 13 having the purpose of generating the aforementioned synchronization signal through which the sorting pattern is transmitted.

In particular, transmitter 11 receives the data generated by the data source 10 and supplies said data encoded in the way described above, together with the corresponding sorting patterns, which are supplied to the FIFO register 12, which accumulates them progressively before their introduction onto the bus 2.

Data accumulated in the FIFO register 12 are sent onto the bus 2 one at a time upon command of a read enable logic signal RE generated by the finite state machine 13 on the basis of the data and of the corresponding sorting patterns supplied thereto.

In particular, the finite state machine 13 has a number of internal states equal to the number of possible sorting patterns, and each of its internal states is uniquely associated to a respective sorting pattern. The finite state machine 13 compares each sorting pattern that it receives from the transmitter 11 with its own internal state and, when these coincide, generates the aforementioned logic synchronization signal Sync, which is supplied to the receiver via a respective additional line of the bus, and through which the receiver recovers the sorting pattern used in transmission.

Upon detection of the coincidence between the sorting pattern to be transmitted and its own internal state, the finite state machine 13 generates the aforesaid read enable signal RE that controls the FIFO register 12 so as to extract from this a new datum and enter it on the bus.

In addition to this, the FIFO register 12 generates a logic empty register signal Empty, indicating the fact that the FIFO register 12 does not contain data to be transmitted, and a logic transmission disabling signal Busy, indicating the fact that the FIFO register 12 is full and is hence no longer capable of accepting input data.

The empty register signal Empty is supplied to the finite state machine 13 with the purpose of interrupting the generation of the synchronization signal Sync when the FIFO register 12 is empty, whilst the transmission disabling signal Busy is supplied both to the data source 10 and to the transmitter 11 for interrupting generation of new data to be transmitted and consequently interrupting supply of new data to the input of the FIFO register 12.

As regards, instead, the circuit architecture at the reception end, Figure 6 illustrates a finite state machine 14 having the function of recovering the sorting pattern used in transmission, and a receiver 15 having the purpose of decoding the data transmitted using the recovered sorting pattern.

In particular, the finite state machine 14 receives the synchronization signal Sync transmitted on the additional line of the bus and the empty register signal Empty and supplies the sorting pattern used in transmission and the read enable signal RE.

In particular, in a way similar to the finite state machine 13, the finite state machine 14 has a number of internal states equal to the number of possible sorting patterns, and each of its internal states is uniquely associated to a respective sorting pattern. In particular, the association between internal state and sorting pattern is exactly identical to that of the finite state machine 13.

In addition, the finite state machine 14 is synchronized with the finite state machine 13, *i.e.*, the internal states of the two finite state machines 13 and 14 coincide at each instant of time.

Thanks to the synchronization and to the identity of association between internal states and sorting patterns of the two finite state machines 13, 14, at the moment of switching of the synchronization signal Sync the finite state machine 14 determines the sorting pattern associated to its internal state at that precise instant in time, a pattern that is exactly identical to the one transmitted.

The synchronization between the two finite state machines is guaranteed by supplying to both the same clock signal CK.

The receiver receives the data transmitted, the sorting pattern recovered, and the read enable signal RE, and supplies the data transmitted. In particular, the data transmitted are received, decoded using the sorting pattern recovered, and then supplied on the output of the receiver only after appropriate enabling provided by the read enable signal RE, which is generated locally by the finite state machine 14.

As mentioned previously, the finite state machines 13 and 14 are designed so as to have a number of internal states equal to the number of possible sorting patterns, and the evolution from one state to the next occurs with a frequency that is determined as a function of the frequency with which the data are sent onto the bus, taking into account the considerations that emerge from what follows.

In the first place, the clock frequency at which the finite state machine 13 operates internally cannot be too high with respect to the one at which the data are introduced onto the bus in so far as such a high frequency is not physically available in the chip in which said architecture is made, and cannot be too low in so far as the time spent for transmitting the data would be unacceptably long.

In particular, if we define $f_{FSM} = f_{bus} \cdot M$, where f_{FSM} represents the clock frequency of the finite state machine 13, f_{bus} represents the clock frequency of the bus, and M represents the number of attempts that are made in one bus cycle for identifying the sorting pattern, the bus cycle being defined as the interval of time elapsing between two successive introductions of data on the bus, and if after each recognition of a sorting pattern the finite state machine 13 is brought into the initial reset state, then the maximum time spent for identification of a sorting pattern is equal to N/M bus cycles.

The fact that the identification of the optimal sorting pattern of a datum involves N/M bus cycles leads to an inevitable delay in transmission of the datum on the bus, so that, in order not to have to interrupt the data source 10 and the transmitter 11, the FIFO register 12 is used, which accumulates the data as these are supplied by the transmitter 11 and enters them one by one on the bus once the corresponding sorting pattern has been identified.

This means that the drastic reduction to one of the transmission lines on which the sorting patterns are transmitted may be obtained only at the expense of a delay in transmission introduced by the FIFO register 12.

The FIFO register 12 is physically unable to have an infinite storage capacity, so that it is necessary to take into account also the likelihood of inoperativeness. Consequently, when the FIFO register 12 is full, the transmission disabling signal Busy is generated, which interrupts precisely the transmission of the data.

Consequently, the technique according to the invention is particularly advantageous and presents good performance in the case where the transmission on the bus is of the burst type (for example AMBA BUS and PCI BUS) and the FIFO register is sized according to the width of the burst.

Simulations conducted by the present applicant have demonstrated that in order to manage the probability of inoperativeness efficiently, the FIFO register 12 may conveniently be built using a process of birth and death of a

Markov chain, which precisely makes it possible to determine *a priori* the probability of inoperativeness of the FIFO register 12 on the basis of the statistical information corresponding to the traffic.

Furthermore, in order to reduce the switching activity on the additional line of the bus through which the synchronization signal Sync containing the information regarding the sorting pattern used in transmission is supplied to the receiver, the coincidence between the internal state of the finite state machine 13 and the sorting pattern to be transmitted, which is supplied by the transmitter 11, is signalled to the finite state machine 14 at the reception end by complementing the synchronization signal Sync, which involves a single switching on the bus (single switching edge of the signal).

In addition, the finite state machines 13, 14 can go into the initial reset state either cyclically, after assuming all the possible internal states, or else after each recognition of a sorting pattern. In particular, the resetting of the finite state machines 13, 14, which can be achieved using precisely the synchronization signal Sync, which, as has been said, is complemented after each recognition of a sorting pattern, enables a further reduction in the transmission delay and is particularly advantageous in the case where the most probable sorting patterns, determined by a preventive analysis of the traffic, are associated to the first internal states of the finite state machines 13, 14.

The reduction of the transmission delay that can be obtained by resetting the finite state machines 13, 14 after each recognition of a sorting pattern is highlighted in Figures 7a and 7b, which show the synchronization signal Sync, the internal states of the finite state machines 13, 14 (considering finite state machines with four internal states S0, S1, S2 and S3), and the optimal sorting pattern transmitted when the finite state machines 13, 14 go into the initial reset state after assuming all the possible internal states (Figure 7a) and, respectively, when the finite state machines 13, 14 go into the initial reset state after each recognition of a sorting pattern (Figure 7b).

From a comparative analysis of the two figures, it is possible immediately to note how, given the same clock cycles, the number of transitions of the synchronization signal is greater when resetting of the finite state machines occurs automatically after each recognition of a sorting pattern as compared to when the resetting of the finite state machines occurs after these have assumed all the possible internal states, said greater number of transitions of the synchronization signal Sync being an index of faster transmission.

Figure 8 illustrates a possible variant of the circuit architecture, at the transmission end, illustrated in Figure 5.

In particular, unlike what is illustrated in Figure 5, where the FIFO register 12 is arranged downstream of the transmitter 11 and stores temporarily both the data supplied by the transmitter 11 and the sorting patterns associated thereto, according to the variant illustrated in Figure 8 the FIFO register 12 is arranged between the data source 10 and the transmitter 11 and stores the data supplied by the data source 10.

In this way, since it is no longer necessary to store the sorting patterns, the FIFO register 12 has a storage capacity considerably smaller than the one present in the embodiment of Figure 5, with consequent reduction in area occupied on the silicon.

This variant requires however, that there be synchronization between the data and the sorting patterns transmitted by the transmitter and the generation of the synchronization signal Sync by the finite state machine 13 so as to enable a correct recognition of the data transmitted at the reception end.

This may be obtained in two ways: either envisaging, as illustrated in Figure 8, an output latch 16 arranged downstream of the transmitter 11, which is loaded with the datum to be transmitted and the corresponding sorting pattern only when the internal state of the finite state machine 13 coincides with the sorting pattern to be transmitted; or else setting the transmitter 11 in stand-by until the

internal state of the finite state machine 13 coincides with the sorting pattern to be transmitted.

In the embodiment illustrated in Figure 8, loading of the latch is enabled directly by the finite state machine 13 through the read enable signal RE that is generated by the latter when its internal state coincides with the sorting pattern to be transmitted, whilst in the other embodiment, not illustrated in Figure 8, the transmitter 11 is set in stand-by by the finite state machine 13 until the internal state of the latter coincides with the sorting pattern to be transmitted.

Also in this embodiment, moreover, the FIFO register 12 generates at output the empty register logic signal Empty for the finite state machine 13, which indicates the fact that the FIFO register 12 does not contain data to be transmitted, and the transmission disabling logic signal Busy for the data source 10, which indicates the fact that the FIFO register 12 is full and is thus no longer able to accept input data.

Figures 9 and 10 illustrate the circuit architecture, at the transmission end and at the reception end, with which synchronization of the sorting pattern is obtained according to a second embodiment of the present invention, which is particularly advantageous when there are additional transmission lines on the bus that are freely usable.

In particular, unlike the first embodiment of the present invention previously described with reference to Figures 5 and 6, the second embodiment does not entail the use of a FIFO register in transmission, and its elimination is made possible by the adoption of a parallel architecture that exploits said additional free transmission lines in combination with the use of different sorting patterns.

In particular, according to what is illustrated in Figure 9, where parts that are identical to the ones appearing in Figure 5 are identified with the same reference numbers, instead of the FIFO register 12, *N/M* finite state machines are used, in which *N* and *M* have the meanings indicated above, *i.e.*, *N* is the number

of possible sorting patterns and *M* is the number of times that the clock frequency of each finite state machine 13 is higher than the clock frequency of the bus.

The finite state machines receive the sorting pattern supplied by the transmitter 11, and each of them supplies a corresponding synchronization signal, designated by Sync1, Sync2, Sync3 and Sync4, which is sent to the reception end through a corresponding additional line of the bus.

Furthermore, the finite state machines have the same number of internal states smaller than the number *N* of possible sorting patterns, and associated to the internal states of each finite state machine is a subset of the possible sorting patterns that are distinct and disjoint with respect to the subsets associated to the other finite state machines; *i.e.*, the subsets of the possible sorting patterns of the finite state machines do not contain elements in common.

For convenience of illustration and description, Figure 9 illustrates, by way of non-limiting example, the case where the number of possible sorting patterns is $N = 16$ and the finite state machines work at a clock frequency four times greater than the clock frequency of the bus, *i.e.*, $M = 4$, it remaining understood that what has been said as regards this specific example is of altogether general application.

The finite state machines necessary for implementing the second embodiment are consequently four ($N/M = 4$), which are designated in Figure 8 by 13.1, 13.2, 13.3. and 13.4, and the corresponding synchronization signals by Sync1, Sync2, Sync3 and Sync4.

Each of the four finite state machines 13.1-13.4 has a number of internal states equal to four, and associated to the four internal states of each finite state machine are four of the possible sorting patterns. Furthermore, the four subsets of sorting patterns associated to the internal states of the four finite state machines 13.1-13.4 do not contain elements in common.

In this way, when the transmitter 11 supplies the datum to be transmitted together with the corresponding sorting pattern, each of the four finite

state machines 13.1-13.4 compares the sorting pattern received with its own internal state, and given that the clock frequency of the finite state machines 13.1-13.4 is four times higher than the clock frequency of the bus, at the end of one bus cycle the sorting pattern to be transmitted will have been recognized, *i.e.*, the internal state of one of the finite state machines 13.1-13.4 will certainly coincide with the sorting pattern to be transmitted.

Upon detection of the coincidence between the sorting pattern to be transmitted and its own internal state, the finite state machine 13.1-13.4 in question generates the corresponding synchronization signal Sync1-Sync4 that is sent to the reception end.

Furthermore, the finite state machines 13.1-13.4 can go into the initial reset state, either cyclically after assuming all the possible internal states or else after each recognition of a sorting pattern. In order to ensure that all the finite state machines 13.1-13.4 go into the initial reset state after each recognition of a sorting pattern, the synchronization signal Sync1-Sync4 generated by a finite state machine 13.1-13.4 is supplied also to all the other finite state machines 13.1-13.4, which are reset upon detection of the switching of any one of the synchronization signals Sync1-Sync4.

As regards the reception end, as illustrated in Figure 10, in which parts that are identical to the ones appearing in Figure 6 are identified with the same reference numbers, there are present *N/M* finite state machines, each of which receives at input a corresponding synchronization signal Sync and supplies at output a corresponding sorting pattern and a corresponding read enable signal RE, and an arbiter which receives the sorting patterns and the read enable signals RE generated by the finite state machines and which supplies to the receiver the sorting pattern and the read enable signal RE generated by the finite state machine involved in the recognition of the sorting pattern.

Furthermore, the finite state machines at the reception end are functionally identical to the finite state machines at the transmission end; *i.e.*, they

have the same number of internal states as the finite state machines at the transmission end, and associated to each one of them is the same subset of sorting patterns that is associated to the finite state machine at the transmission end, from which it receives the synchronization signal.

With reference to the example illustrated in Figure 9, the four finite state machines at the reception end are designated respectively by 14.1, 14.2, 14.3 and 14.4, the sorting patterns supplied by the latter are respectively designated by Pattern1, Pattern2, Pattern3 and Pattern4, the read enable signals generated by them are respectively designated by RE1, RE2, RE3 and RE4, and the arbiter is designated by 17.

When one of the four finite state machines 14.1-14.4 verifies that switching of the corresponding synchronization signal Sync1-Sync4 has occurred, it recovers the sorting pattern transmitted in the bus cycle in which said switching has occurred and generates the sorting pattern transmitted and the corresponding read enable signal, which, via the arbiter 17, are supplied to the receiver 15, which uses them for decoding the data transmitted.

Also in this case the finite state machines 14.1-14.4 can go into the initial reset state either cyclically after assuming all the possible internal states or else after each recognition of a sorting pattern. To ensure that all the finite state machines 14.1-14.4 go into the initial reset state after each recognition of a sorting pattern, each of the finite state machines 14.1-14.4 at the reception end receives all the synchronization signals generated by the finite state machines 13.1-13.4 at the transmission end so as to reset itself upon detection of switching of any one of the synchronization signals.

In particular, for this purpose each finite state machine 14.1-14.4 observes the following rules:

- if its own synchronization signal switches, the sorting pattern is recognized, its own read enable signal goes to the high logic state, and the finite state machine goes into the reset state;

- if one of the other synchronization signals switches, its own read enable signal goes to the low logic state, and the finite state machine goes into the reset state; and

- if no synchronization signal switches, the finite state machine goes into the next internal state.

It is further deemed useful to point out that, with the architecture according to the second embodiment of the present invention (illustrated in Figures 9 and 10), the switching activity of the additional lines of the bus used for transmitting the synchronization signals is equal to the one that exists in the architecture according to the first embodiment of the present invention (illustrated in Figures 5 and 6) and in the corresponding variant (illustrated in Figure 8), in which just one finite state machine is used at the transmission end, and a corresponding finite state machine is used at the reception end in so far as in the second embodiment just one of the finite state machines at the transmission end recognizes the sorting pattern and generates the corresponding synchronization signal in each bus cycle.

The following table sums up the improvement that the present invention makes available in terms of reduction of switching activity *SA* as compared to the known art in a specific case of transmission, on a bus with thirty lines and with a cluster depth equal to six, of files of a different type (Latex, Spice, GCC, JPEG, MP3 and AVI).

Type of file	Classic technique	New technique
LaTeX	26.48 %	37.4 %
Spice	25.84 %	36.59 %
Gcc	25.96 %	36.40 %
Jpeg	15.92 %	27.03 %
Mp3	15.53 %	27.06 %
Avi	35.00 %	36.06 %

From an a examination of the characteristics of the present invention the advantages that it makes possible are evident.

In particular, it makes it possible to reduce to one the number of the additional transmission lines necessary for transmitting the sorting patterns, with a consequent drastic reduction in the bus switching activity of up to 35% with respect to a non-encoded transmission.

Furthermore, as has been said at the beginning of the present description and as will immediately emerge clearly to the reader from the foregoing description, the inventive idea underlying the present invention is of altogether general application and can be used for transmitting data of any nature on a single line.

In fact, to transmit an *n*-bit datum it is sufficient to generate in succession all the possible combinations of *n* bits, compare the *n*-bit datum to be transmitted with the combinations of *n* bits generated, generate an identity signal upon detection of the coincidence between the *n*-bit datum to be transmitted and one of the combinations of *n* bits generated, and transmit the coincidence signal on the single line.

In reception, instead, it is necessary to generate the same succession of combinations of *n* bits generated in transmission, since the successions of combinations of *n* bits generated in transmission and in reception have to be synchronized with one another, and identify the combination of *n* bits generated at the instant of reception of the identity signal transmitted on the single line, the combination of *n* bits generated at the instant of reception of the identity signal being nothing other than the datum to be transmitted.

It will moreover be appreciated that the present invention is suitable for being implemented to particular advantage in the form of a computer product that can be loaded into a memory (typically a set of registers) of a processor associated to the bus and that comprises portions of software code which, when

the computer product is run on the aforesaid processor, perform the steps of the methods according to the invention.

Finally, it is clear that modifications and variations can be made to what has been described and illustrated herein, without thereby departing from the sphere of protection of the present invention, as defined in the attached claims.